SNMP Management in a Distributed Software Router Architecture

SNMP Management in a Distributed Software Router Architecture

Andrea Bianco, Robert Birke, Fikru Getachew Debele, Luca Giraudo
Dip. di Elettronica, Politecnico di Torino, Italy, Email: {last name}@tlc.polito.it

***Abstract*—Multi-stage software router architectures permit to overcome several limitations of single-stage software routers, allowing to expand the number of available interfaces and to increase the overall throughput. However, a multi-stage software router, despite being composed by several internal elements, must externally appear as a single device. A control protocol called DIST was defined to solve this problem from the control plane point of view for a previously proposed multi-stage architecture. In this paper, we tackle the same problem from the network management point of view. We define a management architecture and a manager-agent communication model to coordinate the information residing on the single elements of the multi-stage router to present a unified view to the external network management station issuing SNMP requests. We analyze the different variable types contained in the SNMP MIB-II and divide them into different categories depending on how the response to a SNMP request is compiled. The handling methods used to create the proper response to SNMP request for the different types of MIB variables are described. Analytical computations show that the proposed management architecture does not affect the multi-stage software router scalability.**

I. Introduction

Networking equipments, and routers in particular, are characterized by the development of proprietary architectures. This situation yields to high cost in terms of both equipment and training, because network administrators need to manage different vendor devices or they are forced to a single vendor scenario. This situation drove network researchers to identify software routers (SRs) as an appealing alternative to proprietary devices. SRs are based on personal computers (PCs) running open-source network application software like Linux, Click modular router or XORP [1]–[3]. The main benefits of SRs include: wide availability of multi-vendor hardware and documentation on their architecture and operation, low cost and continuous evolution driven by the PC market's economy of scale. Furthermore, open source SRs provide the opportunity to easily modify the router operation, resulting in flexible and configurable routers. Proprietary network devices often lack programmability and flexibility.

Criticisms to single PC-based SRs are focused on limited performance, software instability, lack of system support, scalability issues, and lack of functionalities. Performance limitations can be compensated by the natural evolution of the performance of the PC architecture. Current PC-based routers and switches can sustain traffic load in the range 1-5 Gbit/s [4], [5], which is enough for a large number of applications. However, high-end performance and large size devices cannot be easily obtained on a single PC.

To overcome these single PC-based router limitations, a multi-stage architecture (shown in Fig. 1) has been suggested in [6]. The multi-stage architecture exploits classical PCs as elementary switching elements to build large high-performance SRs. The proposed architecture has three stages: the layer 2 front-end load balancers (LBs), acting as interfaces to the external networks, the layer 3 back-end routers (BRs) providing the IP routing functionality and an interconnection network to connect the two stages. In the current implementation, the interconnection network is simply an Ethernet switch, the load balancers can be either hardware or software based, and the back-end routers are PCs running the Linux IP stack. The multi-stage architecture comprises a control entity, named virtual Control Processor (CP), that manages, controls and configures the whole architecture [7]. The virtual CP hides the internal details of the multi-stage architecture to external network devices and runs on a selected BR. The key advantages of such architecture among others include the ability to: scale the number of interfaces, recover from faults and provide functional distribution which single PC-based routers lacked.

However, these advantages come at the cost of control and management complexity. This complexity problem has partially been addressed from the control plane perspective [7]. Indeed, an internal control protocol named DIST has been developed to: i) configure the architecture; ii) coordinate the routing process among BRs and the load balancing function among LBs; and iii) provide automatic fault recovery mechanisms. Furthermore, DIST dynamically maps the virtual CP on one of the BRs.

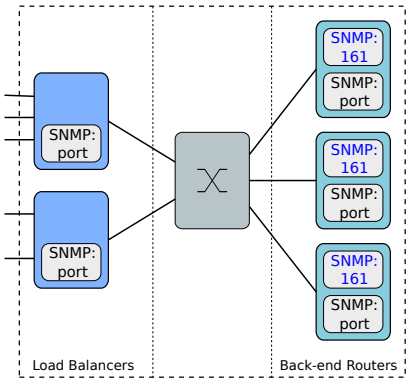

Fig. 1. Multi-stage Software Router Architecture

In this paper we tackle the management plane issue (Sec. II) in the considered multi-stage architecture. A SNMP dual-role entity, named *aggregator* (Sec. III), is introduced to create the multi-stage architecture management base information (Sec. V- VI). Sec. IV describes the required communication model, whereas Sec. VII discusses the scalability of the approach. Finally, the paper is concluded in Sec. VIII.

II. Problem Description

Networks require management procedures to be successfully operated. To reach this goal, network devices keep track of management information, such as cross-traffic, interface status and general system statistics. This information is organized into a management information base (MIB) [8] accessible through different management protocols such as SNMP, NetConf and NetFlow. We focus on SNMP, the most widely used protocol. SNMP is based on a manager-agent model consisting of an SNMP manager, an SNMP agent and a managed device deploying a MIB. The SNMP manager provides the human interface to the management system, while the SNMP agent interfaces the manager and the device MIB.

Managers and agents use SNMP messages for information exchange. SNMP messages comprise request, response and trap messages. Requests can be either *get* or *set* a specific MIB variable. The agent, upon a request reception, sends back a response message with the result of the operation. Trap messages allow the agent to automatically notify the SNMP manager of important events.

Whereas in a single device MIB variables are directly accessible, in a multi-stage architecture they are distributed among different internal elements. This requires additional complexity to collect and/or aggregate the information to compile the response for the manager. Indeed, the management of the multi-stage router requires the provisioning of a unified single-entity management information view to external devices. Therefore, it is necessary to address the problem of mapping and combining the distributed information into an aggregated view. This problem is twofold: i) definition of a communication model to collect the distributed data and ii) mapping the various data to create a single-entity view. An agent residing in the multi-stage software router must coordinate the internal elements and operate on the MIB information distributed among the internal elements to create such a single-entity view.

III. Management Architecture

Fig. 2 depicts the proposed logical architecture. In this architecture, one SNMP dual-role entity, named *aggregator*, coordinates the internal independent SNMP *agents*, and interacts with the external managers issuing SNMP requests. Although the aggregator could run independently, it is integrated in the software modules running on the virtual CP.

LBs redirect any external SNMP requests to this entity. When an SNMP request is received, the aggregator queries the internal SNMP agents to obtain the required MIB information and aggregates the information representing the multi-stage

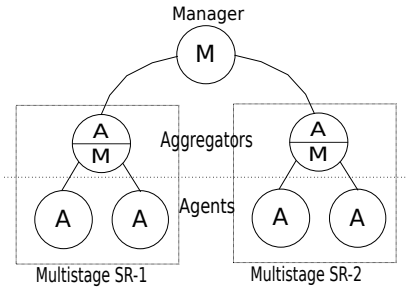

Fig. 2. Management System Used in Multi-Stage Software Router: logical architecture.

architecture. Thus, the aggregator is a dual-role entity that acts as an agent for external managers and as a manager for internal agents.

In our implementation, whereas the LBs and the switch (may) run an SNMP agent, BRs host both the *aggregator* and the *agent* functionalities. More precisely, each BR runs two instances of an SNMP process: an aggregator (listening on the standard SNMP port to be reachable from external managers) and a standard agent listening on a different, configurable, port, used for internal communication. Although all BRs are listening on the standard SNMP port, only one aggregator handles the external requests, because LBs explicitly forward SNMP request to the *active* aggregator which resides on the BR designated as the virtual CP by the DIST protocol. This scheme provides fast fault recovery for the multi-stage architecture in case the *active* aggregator fails. The take over procedure is taken care by the DIST protocol. When a failure is detected DIST elects a new Virtual-CP and reconfigures the LBs to properly redirect SNMP traffic.

Observe that not all the internal elements may be SNMP-capable. Whereas BRs , being based on Linux PCs, can be assumed to be SNMP-capable, LBs, especially if hardware based, might not run an SNMP agent. Therefore, we assume two classes of LBs: SNMP-capable and SNMP-incapable. This assumption affects the way in which the aggregator collects and computes MIB variables, as detailed in Sec. VI.

To ease information sharing among potential aggregators (which is needed for a quick takeover in case of failure) and communication with the DIST protocol entity, the architecture also comprises a database which stores configuration information of the internal elements and most MIB counters. If the LB is not SNMP-capable, a minimal set of interface statistics, namely the received/transmitted bytes/packets and the link speed information are also saved in the database.

The management architecture was implemented and verified in a test-bed. The prototype is based on a customized version of Net-SNMP [9] and MySQL DBMS in addition to the software required to implement the multi-stage architecture.

IV. Manager-agent communication model

The standard communication scenario used in SNMP [10] is defined for a single device which has all the information in the local MIB. However, since in the multi-stage architecture

the aggregator does not have the whole information locally available, a modification to the standard SNMP manager-agent communication model is required.

Fig. 3 shows the modified manager-agent communication model. The dashed box includes the required extensions to deal with the multi-stage architecture. Upon request reception, the aggregator agent checks if the requested variable is locally available. If yes, it responds with the current available value. Otherwise, the aggregator manager sends SNMP requests to the appropriate internal element(s), collects the response(s) received within a given timeout (in our implementation set to one second) and, if required, aggregates the data. Finally, the aggregator agent answers to the original external SNMP request with this value.

If multiple variables must be collected from the internal elements, a response to the external manager's request is sent on the basis of the available information at a given time, even if some responses from internal agents are not available yet. For those elements which did not respond for whatever reason, the aggregator uses, if available, the corresponding variable value saved in the database at the previous successful request. For counter type MIB variable, in the next request, if previously dead agent comes back to service, the aggregator checks the occurrence of any variable re-initialization: if found, the old value contained in the database and the newly available counter value are summed up to mask the discontinuity. This compensation guarantees that counters are kept monotonically increasing. Violating the monotonicity behavior of counters would be disturbing for the external management software, because these values are typically used to compute temporal trends.

V. Multi-stage router MIB

In the MIB definition of our distributed architecture, we mainly consider, among all variables defined in the MIB-II tree, the system, the interface (IF) and the IP group objects for simplicity reason. These variables are grouped into two categories, based on how the aggregator computes the response:

a) Global Variables: E.g. the routing table (*ipRouteTable*), the system up time (*sysUpTime*) or the system name (*sysName*). Since they do not depend on a specific internal element, they are stored in the database to ease information sharing. A response to an external SNMP request for these variables translates into a simple query to the database, which might be populated by the aggregator itself or by the DIST daemon depending on the specific information. For example, the system name is provided by the aggregator, while the routing information is updated by DIST.

b) Collected Variables: This category comprises all the variables requiring collection of data from one or more internal agents, e.g., interface information. A further division is between *specific* and *aggregated* variables. *Specific* variables can be fetched through a single request to a specific internal element. This group comprises all the variables containing specific properties of an internal element, e.g., the link status

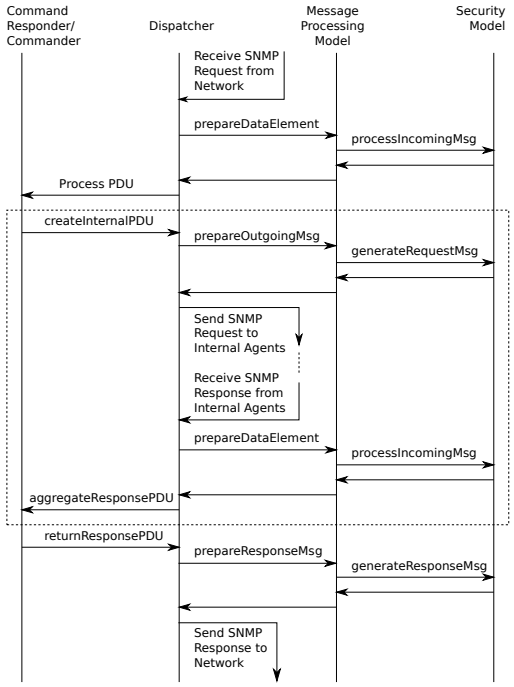

Fig. 3. Modified incoming request scenario diagram for the multi-stage software router

(*ifOperStatus*) or the link speed (*ifSpeed*). *Aggregate* variables need instead multiple queries to different internal agents and require some kind of data aggregation. For instance, the total number of received packets at the IP layer (*ipInReceives*) or the discarded packets at the interface (*ifInDiscards*) are computed using counters from several internal elements.

VI. Single-entity management information view

Global and *collected specific* variables are easy to handle: a simple request forwarding either to the database manager or to an internal agent is needed. Thus, we focus on how to compute the more complex *aggregated* variables (e.g. *IF* and *IP* counters).

Fig. 4 shows the main counters involved during packet forwarding, both in a single device router and in the multi-stage software router. The challenge is to define a mapping between the counters on the left, representing the single-entity view, and the counters on the right distributed over the three stages of the multi-stage architecture. Where ambiguity might exist, we use an over-line to indicate the mapped computed variables and the superscript LB and BR for counters at LB or BR interface respectively. Furthermore, for simplicity, we define $ifInPkts$ as the sum of both unicast ($ifInUcastPkts$) and non-unicast ($ifInNUcastPkts$) packets.

1) $ifInOctets$*,* $ifInErrors$ *and* $ifInUnknownProtos$*:* These variables count respectively the number of received bytes, the number of discarded packets due to errors and unknown/unsupported protocols. These counters are interface specific and, therefore, simply treated as collected specific variables.

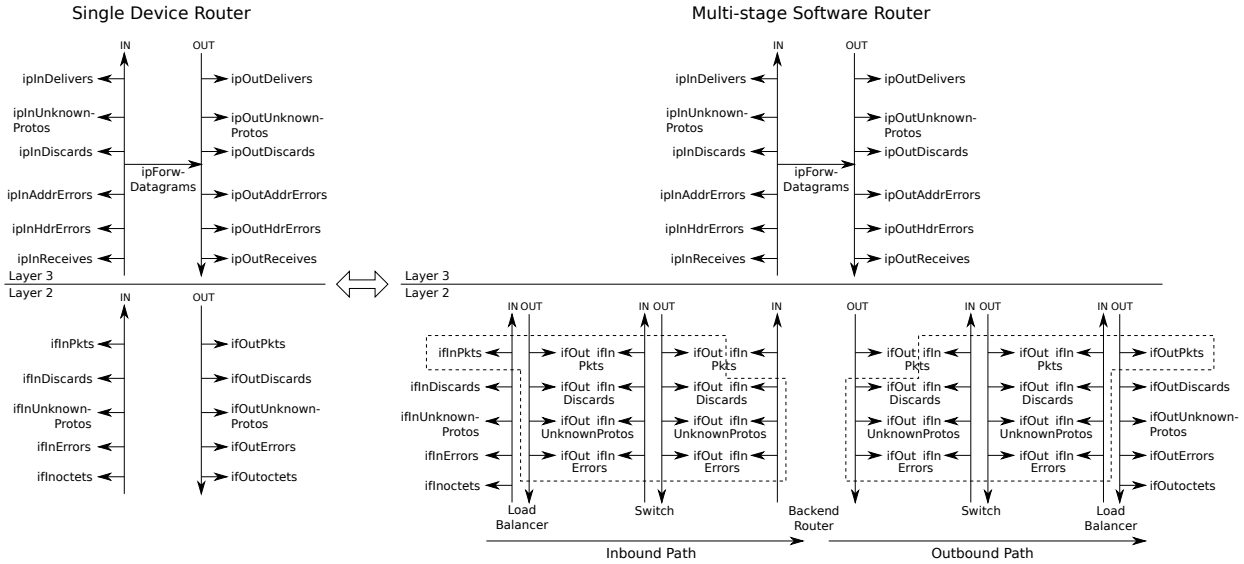

Fig. 4. Main IF and IP counters involved in packet forwarding for a single-stage router (right) and the multi-stage software router (left). For simplicity, $ifInPkts$ represents both $ifInUcastPkts$ and $ifInNUcastPkts$.

As introduced in Sec. III, the computation of MIB variables may be difficult because some internal elements may be non SNMP-capable. For this reason, we consider three different cases for LBs:

- SNMP-capable LBs: SNMP messages are used;
- SNMP-incapable, DIST-capable LBs: The existing control plane is extended to transport minimal traffic statistics (e.g. packet and byte counters);
- SNMP-incapable and DIST-incapable LBs: Data collecting is not possible. Counters are approximated using the information available at the BRs. Thus, some variables, e.g., $ifInErrors$, are not available, because these events occur at LBs' interfaces only.

Similar considerations apply to the outgoing counterpart variables: $ifOutOctets$, $ifOutErrors$ and $ifOutUnknownProtos$.

2) $\overline{ifInDiscards}$*:* As defined in the RFC 1213 [8], the variable $ifInDiscards$ counts the packets which, even if correct at reception time, are discarded by the device for any reason. We use this definition to compute all packets lost while traversing the internal network of the multi-stage architecture. However, it is not possible to track the exact path (and thereof the exact counters involved) of each packet within the multi-stage architecture, due to the unpredictable decision of the load balancing scheduler. Hence, we define D_i as the share of packets internally discarded for interface i.

D_i is computed as the difference of the correctly received packets at the input interface of the LB ($ifInPkts_i^{LB}$) and the sum of correctly received packets at all BRs interfaces R^{BR}, weighted by w_i, the percentage of traffic received at interface i. R^{BR} and w_i are computed as:

$$R^{BR} = \sum_{j=1}^{M}(ifInPkts_j^{BR}) \quad (1)$$

$$w_i = \frac{ifInOctets_i^{LB}}{\sum_{k=1}^{N} ifInOctets_k^{LB}} \quad (2)$$

where M is the total number of BRs interfaces, and N the total number of external LBs interfaces. Thus,

$$D_i = ifInPkts_i^{LB} - w_i R^{BR} \quad (3)$$

$$\overline{ifInDiscards_i} = ifInDiscards_i^{LB} + D_i \quad (4)$$

The above formulas make the implicit assumption that the loss probability is the same on all internal paths, but it has the nice property of being completely independent of the internal load balancing scheme adopted. Thanks to this property, the same procedure can also be applied on the reverse path to compute $\overline{ifOutDiscard_i}$ without knowing the result of the routing operation.

In case of SNMP-incapable and DIST-incapable LB, $\overline{ifInDiscards_i}$ is directly approximated by D_i, replacing $ifInOctets$ and $ifInPkts$ in Eq. (1)-(4) with the received bytes $rxBytes$ and received packets $rxPkts$ statistics stored in the database by DIST.

3) $\overline{ifInPkts}$*:* $ifInPkts$ is the sum of all the corresponding counters at the BRs weighted by w_i.

$$\overline{ifInPkts_i^{LB}} = w_i(\sum_{j=1}^{M} ifInPkts_j^{BR}) \quad (5)$$

For SNMP-incapable and DIST-incapable LB the same substitution as for $ifInDiscards$ apply.

4) IP counters: The IP counters are located only at the BRs. The mapping consists of the sum of all the corresponding IP counters at the BRs. For instance, $\overline{ipInReceives}$ is computed as:

$$\overline{ipInReceives} = \sum_{j=1}^{M} ipInReceives_j^{BR} \quad (6)$$

5) $sysUpTime$: A special mention is needed for the $sysUpTime$ variable, because this information is used as a time reference for the other variables by the external management software, to plot temporal graphs. $sysUpTime$, is a global variable used to store the elapsed time since the management system was running. Given that the aggregator can run on different BRs at different time, it is important that the $sysUpTime$ is not related to a specific instance of the aggregator, but rather tied to the up time of the whole architecture. To achieve this, the first aggregator stores the reference startup time into the database. When an aggregator fails and another takes over, the start up information remains the same. The $sysUpTime$ is re-initialized only if all the BRs fail, i.e., when the multi-stage router fails.

VII. Scalability Analysis

The use of a centralized aggregator has the advantage of reduced management complexity. However, scalability issue might arise due to the concentration of SNMP traffic. Therefore, we try to estimate the amount of SNMP traffic internally generated to process an external SNMP request. The worst case scenario is a request for $IfInDiscards$, because it implies the collection of the largest number of variables from the multi-stage architecture (see Eq. (1)-(4)).

As reported in Sec. VI, M is the total number of BR interfaces, whereas N is the total number of external LB interfaces. Eq. (1) requires to collect $2M$ variables, because $IfInPkts$ is the sum of two variables and Eq. (2) requires N variables. Furthermore, three more variables are needed for Eq. (3) and (4). In the worst case, for each variable two SNMP messages (request and response) are required. Typically, the management station repeats the requests periodically to plot temporal graphs and keep device history. Therefore, the amount of management traffic can be computed as:

$$\text{total\_traffic} = \frac{2(2M+N+3)S}{T} \approx \frac{2(2M+N)S}{T} \quad (7)$$

where S is the SNMP message size, typically about 100 bytes for SNMP responses [11], and T is the update period, typically set to about 5 minutes.

Let us now consider two scenarios: i) a medium range edge router with 360 interfaces at 1Gbps (i.e. a mid-range 7600 series Cisco router [12]) and ii) a core router with 16 interfaces at 10Gbps (i.e. a high-end Juniper T series router [13]). Assuming PCs with 1Gbps routing capability and one LB per interface (worst case in terms of generated messages), then for i) $M = 360, N = 360$ and for ii) $M = 160, N = 16$. Even assuming a very aggressive update period of $1s$, the management traffic would be equal to 216 KBytes/s and 67 Kbytes/s respectively for one MIB variable. Even considering tens of MIB variables traced by the management station, the management traffic is negligible with respect to the total forwarding routing capacity, posing no threat to the overall architecture. Furthermore, the above formula overestimates the real internal management traffic, because an SNMP request is smaller than a SNMP response message and, more importantly, it does not consider the possibility of aggregating more variables into the same SNMP message, which would permit to further reduce the number of messages and increase transmission efficiency.

VIII. Conclusions

The multi-stage software router, being a distributed router architecture, requires a coordinated information management to mask the internal structure and to present the architecture to external SNMP managers as a single device.

We defined a management system based on three elements (*managers*, *aggregators* and *agents*) as an extension of the standard SNMP model. The new system consists of a multi-stage distributed MIB and an extended communication model, which define the mechanisms to collect data from distributed elements in a reliable way and to aggregate the data in an unified view. The net-SNMP 5.4.2.1 [9] implementation has been modified and tested in a small scale test-bed and its scalability was assessed through simple load computations for two classical high-end router configuration.

References

[1] "Linux," http://www.kernel.org/.

[2] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[3] M. Handley, O. Hodson, and E. Kohler, "XORP: an Open Platform for Network Research," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 53–57, 2003.

[4] M. Dobrescu, N. Egi, K. Argyraki, B. G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting Parallelism to Scale Software Routers," in *In Proceedings of the 22nd ACM Symposium on Operating Systems Principles*, 2009.

[5] A. Bianco, R. Birke, D. Bolognesi, J. M. Finochietto, G. Galante, and M. Mellia, "Click vs. Linux: Two Efficient Open-Source IP Network Stacks for Software Routers," in *IEEE Workshop on High Performance Switching and Routing*, 2005, pp. 18–23.

[6] A. Bianco, J. M. Finochietto, M. Mellia, F. Neri, and G. Galante, "Multi-Stage Switching Architectures for Software Routers," *IEEE Network, Issue on Advances in Network Systems*, vol. 33, no. 1, pp. 15 – 21, 2003.

[7] A. Bianco, R. Birke, J. Finochietto, L. Giraudo, F. Marenco, M. Mellia, A. Khan, and D. Manjunath, "Control and Management Plane in a Multi-Stage Software Router Architecture," *In Proc. HPSR*, pp. 235–240, 2008.

[8] K. McCloghrie and M. Rose, "RFC 1213 Management Information Base for Network Management of TCP/IP-based internets: MIB-II," 1991, http://www.rfc-editor.org/rfc/rfc1213.txt.

[9] "Net-SNMP," http://net-snmp.sourceforge.net/.

[10] D. Harrington, R. Presuhn, and B. Wijnen, "RFC 3411 An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," 2002, http://tools.ietf.org/html/rfc3411.

[11] C. Pattinson, "A Study of the Behaviour of the Simple Network Management Protocol," *In Proc. of 12th International Workshop on Distributed Systems, Nancy, France*, 2001.

[12] "Cisco 7600 Series Routers," http://www.cisco.com/en/US/products/hw/routers/ps368/index.html.

[13] "T Series Core Routers," www.juniper.net/us/en/local/pdf/datasheets/1000051-en.pdf.